File Upload and Management Application

Technical Report

Prepared by

Abdullah Demirci

23.05.2025

1. Introduction

This section provides an overview of the "File Upload and Management Application" project, including its purpose, scope, and the organization of this document.

1.1. Purpose of the project

The primary purpose of this project is to develop a simple web application that allows users to upload, list, and delete various file types, specifically PDF, PNG, and JPG files. The application serves as a demonstration of full-stack web development skills, encompassing frontend and backend integration, as well as file management capabilities.

1.2. Scope of the project

The project's scope is defined by the following key functionalities:

-User Authentication: Implementation of a secure login and registration system. In this project, login and registration processes were carried out as simple requests.

-File Upload: Enabling users to upload PDF, PNG, and JPG files transfer to a directed directory.

-File Listing: Displaying a list of all uploaded files to the user.

-File Deletion: Providing a mechanism for users to delete their uploaded files.

-Server-Side Storage: Ensuring that uploaded files are securely stored in a designated directory.

The project utilizes Spring Boot for the backend development and React for the frontend and PostgreSQL as a database ,demonstrating a modern web development stack.

1.3. Overview of the document

This report is structured to provide a comprehensive overview of the "File Upload and Management Application." Following this introduction, Section 2 details the requirements specification, categorizing them into functional, non-functional, and domain requirements. Section 3 covers the software design, including the software model, development tools, architectural design, and object identification. Testing procedures are outlined in Section 4, followed by the conclusion in Section 5 and references in Section 6.

2. Requirements Specification

This section details the functional, non-functional, and domain requirements for the "File Upload and Management Application."

2.1. Functional requirements

-The system shall allow new users to register an account with a unique username and password.

-The system shall allow registered users to log in using their credentials.

-The system shall allow authenticated users to upload PDF, PNG, and JPG file types.

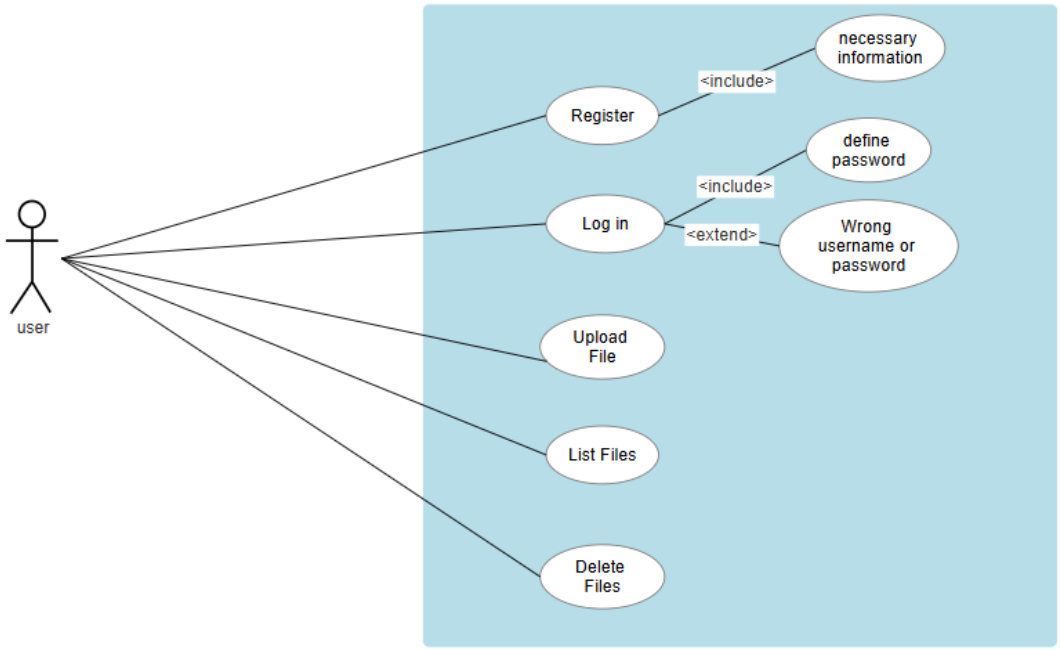
-The system shall display a list of all files uploaded by the current user.

-For each listed file, the system shall display its name, size, and upload date.

-The system shall allow users to delete individual files from the list.

-The system shall provide a logout functionality.

-Uploaded files must be stored persistently designated storage directory.



2.2. Non-functional requirements

-Security: User authentication (login/registration) should be secure, possibly using JWT (may be added to the project in the future) or simple session management.

-Usability: The user interface should be intuitive and easy to navigate for file upload, listing, and deletion.

-Reliability: The system should reliably store and retrieve files without data loss.

-Performance: File uploads and listings should be processed in a timely manner.

-Maintainability: The codebase should be well-structured and documented to facilitate future updates and maintenance.

-Scalability: The architecture should allow for future expansion in terms of user base and file storage capacity.

2.3. Domain requirements

-The system must handle common image formats (PNG, JPG) and document format (PDF).

-File names, sizes, and upload timestamps should be accurately recorded and displayed.

3. Software Design

This section details the software design choices made for the "File Upload and Management Application."

3.1. Software Model

The application follows a client-server architecture. The frontend, developed with React, acts as the client, responsible for user interaction and making requests to the backend. The backend, developed with Spring Boot, serves as the server, handling business logic, file storage, and database interactions.

3.2. Software Development Tools

-Backend: Spring Boot (Java).

-Frontend: React.

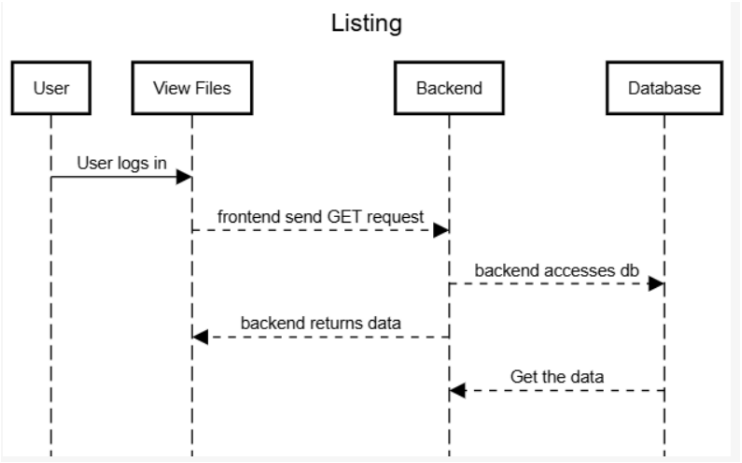
-Database: PostgreSQL Database

-Build Tools: Maven/Gradle for Spring Boot, npm for React.

-Version Control: Git .

3.3. Architectural Design

The application employs a RESTful API architecture for communication between the frontend and backend. The React frontend makes HTTP requests (GET, POST, DELETE) to the Spring Boot backend's API endpoints for user authentication, file uploads, file listings, and file deletions. Files are stored in a designated directory in a database.



3.4. Object identification

Key objects identified in the system include:

-User: Represents a user of the application, with attributes like user mail,username and password.

-File (Document/Image): Represents an uploaded file, with attributes like filename, filesize, uploadDate, and a reference to its storage location.

-Authentication Service: Handles user login, registration, and session management.

-File Management Service: Handles file upload, listing, and deletion operations.

4. Testing

This section outlines the testing strategy for the "File Upload and Management Application."

4.1. Test Plan

Testing will involve a combination of unit tests, integration tests, and end-to-end tests to ensure the application's functionality, reliability, and security.

4.2. Test-design Specification

-Unit Tests: Individual components (e.g., backend services, React components) will be tested to verify their correct behavior.

-Integration Tests: The interaction between different components (e.g., frontend-backend communication, database interactions) will be tested to ensure seamless integration.

-End-to-End Tests: User flows, such as registration, login, file upload, file listing, and file deletion, will be tested from the user's perspective to ensure the entire system functions as expected.

4.3. Test-case specification

-Authentication:

-Valid user registration.

-Invalid user registration .

-Successful login with valid credentials.

-Failed login with invalid credentials.

-Logout functionality.

-File Upload:

-Upload a valid PDF file.

-Upload a valid PNG file.

-Upload a valid JPG file.

-File Listing:

-Verify that newly uploaded files appear in the list.

-Verify that deleted files are removed from the list.

-Verify correct display of file name, size, and date.

-File Deletion:

-Delete an existing file successfully.

5. Conclusion

The "File Upload and Management Application" successfully delivers a fundamental solution for managing files. The implemented features user authentication, file upload, listing, and deletion demonstrate competence in full-stack web development using Spring Boot, React and PostgreSQL. The application adheres to the specified requirements, providing a functional and user-friendly interface for file operations. The chosen technologies provide a strong and scalable foundation for potential future enhancements.

6. References

[1] Spring Boot. (n.d.). Getting Started | Building an Application with Spring Boot. Retrieved from https://spring.io/guides/gs/spring-boot

[2] Marmara Üniversitesi. (n.d.). Project Report Template. Retrieved from https://mimoza.marmara.edu.tr/~samet.tonyali/courses/cse344/projects/project_report_template.pdf

[3] React.dev. (n.d.). Learn React. Retrieved from https://react.dev/learn